ĐỊNH THỜI BIỂU CPU

I Mục tiêu

Sau khi học xong chương này, người học nắm được những kiến thức sau:

- Hiểu các khái niệm cơ bản về định thời
- Hiểu các giải thuật định thời biểu CPU
- Vận dụng một giải thuật định thời cho một hệ thống cụ thể

II Giới thiệu

Định thời biểu là cơ sở của các hệ điều hành đa chương. Bằng cách chuyển đổi CPU giữa các quá trình, hệ điều hành có thể làm máy tính hoạt động nhiều hơn. Trong chương này, chúng ta giới thiệu các khái niệm định thời cơ bản và trình bày các giải thuật định thời biểu CPU khác nhau. Chúng ta cũng xem xét vấn đề chọn một giải thuật cho một hệ thống xác định.

III Các khái niệm cơ bản

Mục tiêu của đa chương là có nhiều quá trình chạy cùng thời điểm để tối ưu hóa việc sử dụng CPU. Trong hệ thống đơn xử lý, chỉ một quá trình có thể chạy tại một thời điểm; bất cứ quá trình nào khác đều phải chờ cho đến khi CPU rảnh và có thể được định thời lại.

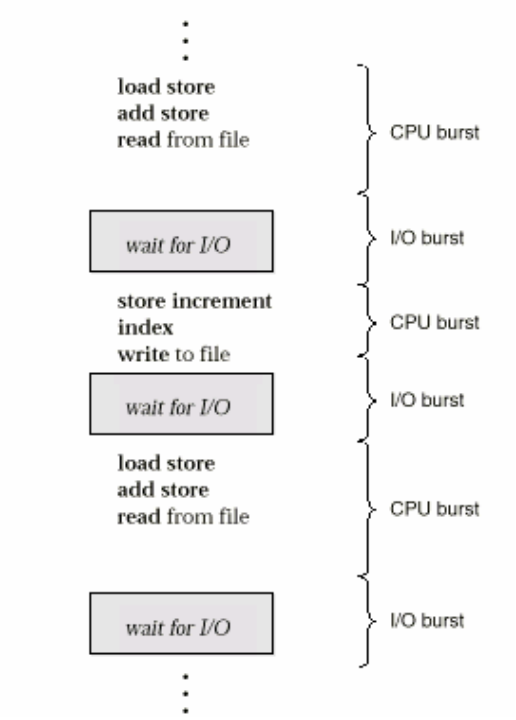
Ý tưởng của đa chương là tương đối đơn giản. Một quá trình được thực thi cho đến khi nó phải chờ yêu cầu nhập/xuất hoàn thành. Trong một hệ thống máy tính đơn giản thì CPU sẽ rảnh rỗi; tất cả thời gian chờ này là lãng phí. Với đa chương, chúng ta cố gắng dùng thời gian này để CPU có thể phục vụ cho các quá trình khác. Nhiều quá trình được giữ trong bộ nhớ tại cùng thời điểm. Khi một quá trình phải chờ, hệ điều hành lấy CPU từ quá trình này và cấp CPU tới quá trình khác.

Định thời biểu là chức năng cơ bản của hệ điều hành. Hầu hết tài nguyên máy tính được định thời biểu trước khi dùng. Dĩ nhiên, CPU là một trong những tài nguyên máy tính ưu tiên. Do đó, định thời biểu là trọng tâm trong việc thiết kế hệ điều hành.

III.1 Chu kỳ CPU-I/O

Sự thành công của việc định thời biểu CPU phụ thuộc vào thuộc tính được xem xét sau đây của quá trình. Việc thực thi quá trình chứa một **chu kỳ** (cycle) thực thi CPU và chờ đợi nhập/xuất. Các quá trình chuyển đổi giữa hai trạng thái này. Sự thực thi quá trình bắt đầu với một **chu kỳ CPU** (CPU burst), theo sau bởi một **chu kỳ nhập/xuất** (I/O burst), sau đó một chu kỳ CPU khác, sau đó lại tới một chu kỳ nhập/xuất khác khác,..Sau cùng, chu kỳ CPU cuối cùng sẽ kết thúc với một yêu cầu hệ thống để kết thúc việc thực thi, hơn là với một chu kỳ nhập/xuất khác, được mô tả như hình IV.1. Một chương trình **hướng nhập/xuất** (I/O-bound) thường có nhiều chu kỳ CPU ngắn. Một chương trình **hướng xử lý** (CPU-bound) có thể có một nhiều chu

kỳ CPU dài. Sự phân bổ này có thể giúp chúng ta chọn giải thuật định thời CPU hợp lý.



Hình 0-1-Thay đổi thứ tự của CPU và I/O burst

III.2 Bộ định thời CPU

Bất cứ khi nào CPU rảnh, hệ điều hành phải chọn một trong những quá trình trong hàng đợi sẵn sàng để thực thi. Chọn quá trình được thực hiện bởi **bộ định thời biểu ngắn** (short-term scheduler) hay bộ định thời CPU. Bộ định thời này chọn các quá trình trong bộ nhớ sẵn sàng thực thi và cấp phát CPU tới một trong các quá trình đó.

Hàng đợi sẵn sàng không nhất thiết là hàng đợi vào trước, ra trước (FIFO). Xem xét một số giải thuật định thời khác nhau, một hàng đợi sẵn sàng có thể được cài đặt như một hàng đợi FIFO, một hàng đợi ưu tiên, một cây, hay đơn giản là một danh sách liên kết không thứ tự. Tuy nhiên, về khái niệm tất cả các quá trình trong hàng đợi sẵn sàng được xếp hàng chờ cơ hội để chạy trên CPU. Các mẫu tin trong hàng đợi thường là khối điều khiển quá trình của quá trình đó.

III.3 Định thời biểu trưng dụng

Quyết định định thời biểu CPU có thể xảy ra một trong 4 trường hợp sau:

- Khi một quá trình chuyển từ trạng thái chạy sang trạng thái chờ (thí dụ: yêu cầu nhập/xuất, hay chờ kết thúc của một trong những quá trình con).
- Khi một quá trình chuyển từ trạng thái chạy tới trạng thái sẵn sàng (thí dụ: khi một ngắt xảy ra)
- Khi một quá trình chuyển từ trạng thái chờ tới trạng thái sẵn sàng (thí dụ: hoàn thành nhập/xuất)

- Khi một quá trình kết thúc

Trong trường hợp 1 và 4, không cần chọn lựa loại định thời biểu. Một quá trình mới (nếu tồn tại trong hàng đợi sẵn sàng) phải được chọn để thực thi. Tuy nhiên, có sự lựa chọn loại định thời biểu trong trường hợp 2 và 3.

Khi định thời biểu xảy ra chỉ trong trường hợp 1 và 4, chúng ta nói cơ chế **định thời không trưng dụng** (nonpreemptive); ngược lại, khi định thời biểu xảy ra chỉ trong trường hợp 2 và 3, chúng ta nói cơ chế **định thời trưng dụng** (preemptive). Trong định thời không trưng dụng, một khi CPU được cấp phát tới một quá trình, quá trình giữ CPU cho tới khi nó giải phóng CPU hay bởi kết thúc hay bởi chuyển tới trạng thái sẵn sàng. Phương pháp định thời biểu này được dùng bởi các hệ điều hành Microsoft Windows 3.1 và bởi Apple Macintosh. Phương pháp này chỉ có thể được dùng trên các nền tảng phần cứng xác định vì nó không đòi hỏi phần cứng đặc biệt (thí dụ, một bộ đếm thời gian) được yêu cầu để định thời biểu trưng dụng.

Tuy nhiên, định thời trưng dụng sinh ra một chi phí. Xét trường hợp 2 quá trình chia sẻ dữ liệu. Một quá trình có thể ở giữa giai đoạn cập nhật dữ liệu thì nó bị chiếm dụng CPU và một quá trình thứ hai đang chạy. Quá trình thứ hai có thể đọc dữ liệu mà nó hiện đang ở trong trạng thái thay đổi. Do đó, những kỹ thuật mới được yêu cầu để điều phối việc truy xuất tới dữ liệu được chia sẻ.

Sự trưng dụng cũng có một ảnh hưởng trong thiết kế nhân hệ điều hành. Trong khi xử lý lời gọi hệ thống, nhân có thể chờ một hoạt động dựa theo hành vi của quá trình. Những hoạt động như thế có thể liên quan với sự thay đổi dữ liệu nhân quan trọng (thí dụ: các hàng đợi nhập/xuất). Điều gì xảy ra nếu quá trình bị trưng dụng CPU ở trong giai đoạn thay đổi này và nhân (hay trình điều khiển thiết bị) cần đọc hay sửa đổi cùng cấu trúc? Sự lộn xộn chắc chắn xảy ra. Một số hệ điều hành, gồm hầu hết các ấn bản của UNIX, giải quyết vấn đề này bằng cách chờ lời gọi hệ thống hoàn thành hay việc nhập/xuất bị nghẽn, trước khi chuyển đổi ngữ cảnh. Cơ chế này đảm bảo rằng cấu trúc nhân là đơn giản vì nhân sẽ không trưng dụng một quá trình trong khi các cấu trúc dữ liệu nhân ở trong trạng thái thay đổi. Tuy nhiên, mô hình thực thi nhân này là mô hình nghèo nàn để hỗ trợ tính toán thời thực và đa xử lý.

Trong trường hợp UNIX, các phần mã vẫn là sự rủi ro. Vì các ngắt có thể xảy ra bất cứ lúc nào và vì các ngắt này không thể luôn được bỏ qua bởi nhân, nên phần mã bị ảnh hưởng bởi ngắt phải được đảm bảo từ việc sử dụng đồng thời. Hệ điều hành cần chấp nhận hầu hết các ngắt, ngược lại dữ liệu nhập có thể bị mất hay dữ liệu xuất bị viết chồng. Vì thế các phần mã này không thể được truy xuất đồng hành bởi nhiều quá trình, chúng vô hiệu hóa ngắt tại lúc nhập và cho phép các ngắt hoạt động trở lại tại thời điểm việc nhập kết thúc. Tuy nhiên, vô hiệu hóa và cho phép ngắt tiêu tốn thời gian, đặc biệt trên các hệ thống đa xử lý.

III.4 Bộ phân phát

Một thành phần khác liên quan đến chức năng định thời biểu CPU là **bộ phân phát** (dispatcher). Bộ phân phát là một module có nhiệm vụ trao điều khiển CPU tới quá trình được chọn bởi bộ định thời biểu ngắn (short-term scheduler). Chức năng này liên quan:

- Chuyển ngữ cảnh
- Chuyển chế độ người dùng
- Nhảy tới vị trí hợp lý trong chương trình người dùng để khởi động lại quá trình

Bộ phân phát nên nhanh nhất có thể, và nó được nạp trong mỗi lần chuyển quá trình. Thời gian mất cho bộ phân phát dừng một quá trình này và bắt đầu chạy một quá trình khác được gọi là thời gian trễ cho việc điều phối (dispatch latency).

IV Các tiêu chuẩn định thời

Các giải thuật định thời khác nhau có các thuộc tính khác nhau và có xu hướng thiên vị cho một loại quá trình hơn một quá trình. Trong việc chọn giải thuật nào sử dụng trong trường hợp nào, chúng ta phải xét các thuộc tính của các giải thuật khác nhau.

Nhiều tiêu chuẩn được đề nghị để so sánh các giải thuật định thời biểu. Những đặc điểm được dùng để so sánh có thể tạo sự khác biệt quan trọng trong việc xác định giải thuật tốt nhất. Các tiêu chuẩn gồm:

- **Việc sử dụng CPU**: chúng ta muốn giữ CPU bận nhiều nhất có thể. Việc sử dụng CPU có thể từ 0 đến 100%. Trong hệ thống thực, nó nên nằm trong khoảng từ 40% (cho hệ thống được nạp tải nhẹ) tới 90% (cho hệ thống được nạp tải nặng).
- **Thông lượng**: nếu CPU bận thực thi các quá trình thì công việc đang được thực hiện. Thước đo của công việc là số lượng quá trình được hoàn thành trên một đơn vị thời gian gọi là **thông lượng** (throughput). Đối với các quá trình dài, tỉ lệ này có thể là 1 quá trình trên 1 giờ; đối với các giao dịch ngắn, thông lượng có thể là 10 quá trình trên giây.
- **Thời gian hoàn thành**: từ quan điểm của một quá trình cụ thể, tiêu chuẩn quan trọng là mất bao lâu để thực thi quá trình đó. Khoảng thời gian từ thời điểm gởi quá trình tới khi quá trình hoàn thành được gọi là **thời gian hoàn thành** (turnaround time). Thời gian hoàn thành là tổng các thời gian chờ đưa quá trình vào bộ nhớ, chờ hàng đợi sẵn sàng, thực thi CPU và thực hiện nhập/xuất.
- **Thời gian chờ**: giải thuật định thời CPU không ảnh hưởng lượng thời gian quá trình thực thi hay thực hiện nhập/xuất; nó ảnh hưởng chỉ lượng thời gian một quá trình phải chờ trong hàng đợi sẵn sàng. **Thời gian chờ** (waiting time) là tổng thời gian chờ trong hàng đợi sẵn sàng.
- **Thời gian đáp ứng**: trong một hệ thống giao tiếp, thời gian hoàn thành không là tiêu chuẩn tốt nhất. Thông thường, một quá trình có thể tạo ra dữ liệu xuất tương đối sớm và có thể tiếp tục tính toán các kết quả mới trong khi các kết quả trước đó đang được xuất cho người dùng. Do đó, một thước đo khác là thời gian từ lúc gởi yêu cầu cho tới khi đáp ứng đầu tiên được tạo ra. Thước đo này được gọi là **thời gian đáp ứng** (response time), là lượng thời gian mất đi từ lúc bắt đầu đáp ứng nhưng không là thời gian mất đi để xuất ra đáp ứng đó. Thời gian hoàn thành thường bị giới hạn bởi tốc độ của thiết bị xuất.

Chúng ta muốn tối ưu hóa việc sử dụng CPU và thông lượng, đồng thời tối thiểu hóa thời gian hoàn thành, thời gian chờ, và thời gian đáp ứng. Trong hầu hết các trường hợp, chúng ta tối ưu hóa thước đo trung bình. Tuy nhiên, trong một vài trường hợp chúng ta muốn tối ưu giá trị tối thiểu hay giá trị tối đa hơn là giá trị trung bình. Thí dụ, để đảm bảo rằng tất cả người dùng nhận dịch vụ tốt, chúng ta muốn tối thiểu thời gian đáp ứng tối đa.

Đối với những hệ thống tương tác (như các hệ thống chia thời), một số nhà phân tích đề nghị rằng sự thay đổi trong thời gian đáp ứng quan trọng hơn tối thiểu hóa thời gian đáp ứng trung bình. Một hệ thống với thời gian đáp ứng phù hợp và có thể đoán trước được quan tâm nhiều hơn hệ thống chạy nhanh hơn mức trung bình nhưng biến đổi cao. Tuy nhiên, gần như không có công việc nào được thực hiện trên các giải thuật định thời biểu CPU để tối thiểu hóa các thay đổi.

Khi chúng ta thảo luận các giải thuật định thời biểu CPU khác nhau, chúng ta muốn hiển thị các hoạt động của chúng. Một hình ảnh chính xác nên thông báo tới nhiều quá trình, mỗi quá trình là một chuỗi của hàng trăm chu kỳ CPU và I/O. Để đơn giản việc hiển thị, chúng ta xem chỉ một chu kỳ CPU (trong mili giây) trên quá trình trong các thí dụ của chúng ta. Thước đo của việc so sánh là thời gian chờ đợi trung bình.

V Các giải thuật định thời

Định thời biểu CPU giải quyết vấn đề quyết định quá trình nào trong hàng đợi sẵn sàng được cấp phát CPU. Trong phần này chúng ta mô tả nhiều giải thuật định thời CPU đang có.

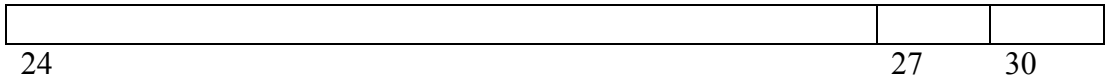
V.1 Định thời đến trước được phục vụ trước

Giải thuật định thời biểu CPU đơn giản nhất là **đến trước, được phục vụ trước** (first-come, first-served-FCFS). Với cơ chế này, quá trình yêu cầu CPU trước được cấp phát CPU trước. Việc cài đặt chính sách FCFS được quản lý dễ dàng với hàng đợi FIFO. Khi một quá trình đi vào hàng đợi sẵn sàng, PCB của nó được liên kết tới đuôi của hàng đợi. Khi CPU rảnh, nó được cấp phát tới một quá trình tại đầu hàng đợi. Sau đó, quá trình đang chạy được lấy ra khỏi hàng đợi. Mã của giải thuật FCFS đơn giản để viết và hiểu.

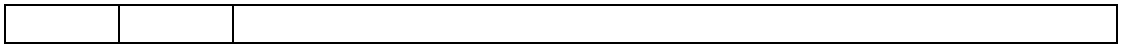
Tuy nhiên, thời gian chờ đợi trung bình dưới chính sách FCFS thường là dài. Xét tập hợp các quá trình sau đến tại thời điểm 0, với chiều dài thời gian chu kỳ CPU được cho theo mini giây.

Quá trình	Thời gian xử lý
P1	24
P2	3
P3	3

Nếu các quá trình đến theo thứ tự P_1, P_2, P_3 và được phục vụ theo thứ tự FCFS, chúng ta nhận được kết quả được hiển thị trong **lưu đồ Gantt** như sau:



Thời gian chờ là 0 mili giây cho quá trình P_1, 24 mili giây cho quá trình P_2 và 27 mili giây cho quá trình P_3. Do đó, thời gian chờ đợi trung bình là (0+24+27)/3=17 mili giây. Tuy nhiên, nếu các quá trình đến theo thứ tự P_2, P_3, P_1 thì các kết quả được hiển thị trong **lưu đồ Gannt** như sau:



Thời gian chờ đợi trung bình bây giờ là (6+0+3)/3=3 mili giây. Việc cắt giảm này là quan trọng. Do đó, thời gian chờ đợi trung bình dưới chính sách FCFS thường không là tối thiểu và có sự thay đổi rất quan trọng nếu các thời gian CPU dành cho các quá trình khác nhau rất lớn.

Ngoài ra, xét năng lực của định thời FCFS trong trường hợp động. Giả sử chúng ta có một quá trình hướng xử lý (CPU-bound) và nhiều quá trình hướng nhập/xuất (I/O bound). Khi các quá trình đưa đến quanh hệ thống, ngữ cảnh sau có thể xảy ra. Quá trình hướng xử lý sẽ nhận CPU và giữ nó. Trong suốt thời gian này, tất cả quá trình khác sẽ kết thúc việc nhập/xuất của nó và chuyển vào hàng đợi sẵn sàng, các thiết bị nhập/xuất ở trạng thái rảnh. Cuối cùng, quá trình hướng xử lý kết thúc chu kỳ CPU của nó và chuyển tới thiết bị nhập/xuất. Tất cả các quá trình hướng xử lý có chu kỳ CPU rất ngắn sẽ nhanh chóng thực thi và di chuyển trở về hàng đợi nhập/xuất. Tại thời điểm này CPU ở trạng thái rảnh. Sau đó, quá trình hướng xử lý sẽ di chuyển trở lại hàng đợi sẵn sàng và được cấp CPU. Một lần nữa, tất cả quá trình hướng nhập/xuất kết thúc việc chờ trong hàng đợi sẵn sàng cho đến khi quá trình hướng xử lý được thực hiện. Có một **tác dụng phụ** (convoy effect) khi tất cả các quá trình khác chờ một quá trình lớn trả lại CPU. Tác dụng phụ này dẫn đến việc sử dụng thiết bị và CPU thấp hơn nếu các quá trình ngắn hơn được cấp trước.

Giải thuật FCSF là giải thuật định thời không trưng dụng CPU. Một khi CPU được cấp phát tới một quá trình, quá trình đó giữ CPU cho tới khi nó giải phóng CPU bằng cách kết thúc hay yêu cầu nhập/xuất. Giải thuật FCFS đặc biệt không phù hợp đối với hệ thống chia sẻ thời gian, ở đó mỗi người dùng nhận được sự chia sẻ CPU với những khoảng thời gian đều nhau.

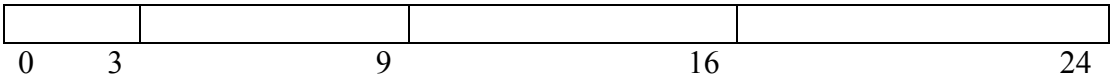
V.2 Định thời biểu công việc ngắn nhất trước

Một tiếp cận khác đối với việc định thời CPU là giải thuật định thời **công việc ngắn nhất trước** (shortest-job-first-SJF). Giải thuật này gán tới mỗi quá trình chiều dài của chu kỳ CPU tiếp theo cho quá trình sau đó. Khi CPU sẵn dùng, nó được gán tới quá trình có chu kỳ CPU kế tiếp ngắn nhất. Nếu hai quá trình có cùng chiều dài chu kỳ CPU kế tiếp, định thời FCFS được dùng. Chú ý rằng thuật ngữ phù hợp hơn là chu kỳ CPU kế tiếp ngắn nhất (shortest next CPU burst) vì định thời được thực hiện bằng cách xem xét chiều dài của chu kỳ CPU kế tiếp của quá trình hơn là toàn bộ chiều dài của nó. Chúng ta dùng thuật ngữ SJF vì hầu hết mọi người và mọi sách tham khảo tới nguyên lý của loại định thời biểu này như SJF.

Thí dụ, xét tập hợp các quá trình sau, với chiều dài của thời gian chu kỳ CPU được tính bằng mili giây:

Quá trình	Thời gian xử lý
P1	6
P2	8
P3	7
P4	3

Dùng định thời SJF, chúng ta định thời biểu cho các quá trình này theo **lưu đồ Gannt** như sau:



Thời gian chờ đợi là 3 mili giây cho quá trình P_1, 16 mili giây cho quá trình P_2, 9 mili giây cho quá trình P_3, và 0 mili giây cho quá trình P_4. Do đó, thời gian chờ đợi trung bình là (3+16+9+0)/4 = 7 mili giây. Nếu chúng ta dùng cơ chế định thời FCFS thì thời gian chờ đợi trung bình là 10.23 mili giây.

Giải thuật SJF có thể là tối ưu, trong đó nó cho thời gian chờ đợi trung bình nhỏ nhất cho các quá trình được cho. Bằng cách chuyển một quá trình ngắn trước một quá trình dài thì thời gian chờ đợi của quá trình ngắn giảm hơn so với việc tăng thời gian chờ đợi của quá trình dài. Do đó, thời gian chờ đợi trung bình giảm.

Khó khăn thật sự với giải thuật SJF là làm thế nào để biết chiều dài của yêu cầu CPU tiếp theo. Đối với định thời dài trong hệ thống bó, chúng ta có thể dùng chiều dài như giới hạn thời gian xử lý mà người dùng xác định khi gởi công việc. Do đó, người dùng được cơ động để ước lượng chính xác giới hạn thời gian xử lý vì giá trị thấp hơn có nghĩa là đáp ứng nhanh hơn. Định thời SJF được dùng thường xuyên trong định thời dài.

Mặc dù SJF là tối ưu nhưng nó không thể được cài đặt tại cấp định thời CPU ngắn vì không có cách nào để biết chiều dài chu kỳ CPU tiếp theo. Một tiếp cận là khác gần đúng định thời SJF được thực hiện. Chúng ta có thể không biết chiều dài của chu kỳ CPU kế tiếp nhưng chúng ta có đoán giá trị của nó. Chúng ta mong muốn rằng chu kỳ CPU kế tiếp sẽ tương tự chiều dài những chu kỳ CPU trước đó. Do đó, bằng cách tính toán mức xấp xỉ chiều dài của chu kỳ CPU kế tiếp, chúng ta chọn một quá trình với chu kỳ CPU được đoán là ngắn nhất.

Chu kỳ CPU kế tiếp thường được đoán như trung bình số mũ của chiều dài các chu kỳ CPU trước đó. Gọi t_n là chiều dài của chu kỳ CPU thứ n và gọi T_{n+1} giá trị được đoán cho chu kỳ CPU kế tiếp. Thì đối với α, với $0 \le \alpha \le 1$, định nghĩa

$$T_{n+1} = \alpha\, t_n + (1- \alpha)\, T_n$$

Công thức này định nghĩa một giá trị trung bình số mũ. Giá trị của t_n chứa thông tin mới nhất; T_n lưu lịch sử quá khứ. Tham số α điều khiển trọng số liên quan giữa lịch sử quá khứ và lịch sử gần đây trong việc đoán. Nếu α=0 thì $T_{n+1}=T_n$ và lịch sử gần đây không có ảnh hưởng (điều kiện hiện hành được đảm bảo là ngắn); nếu α =1 thì $T_{n+1}=t_n$ và chỉ chu kỳ CPU gần nhất có ảnh hưởng (lịch sử được đảm bảo là cũ và không phù hợp). Thông dụng hơn, α=1/2 thì lịch sử gần đây và lịch sử quá khứ có trọng số tương đương. Giá trị khởi đầu T_0 có thể được định nghĩa như một hằng số hay như toàn bộ giá trị trung bình hệ thống. Hình IV.2 dưới đây hiển thị giá trị trung bình dạng mũ với α=1/2 và T_0=10.

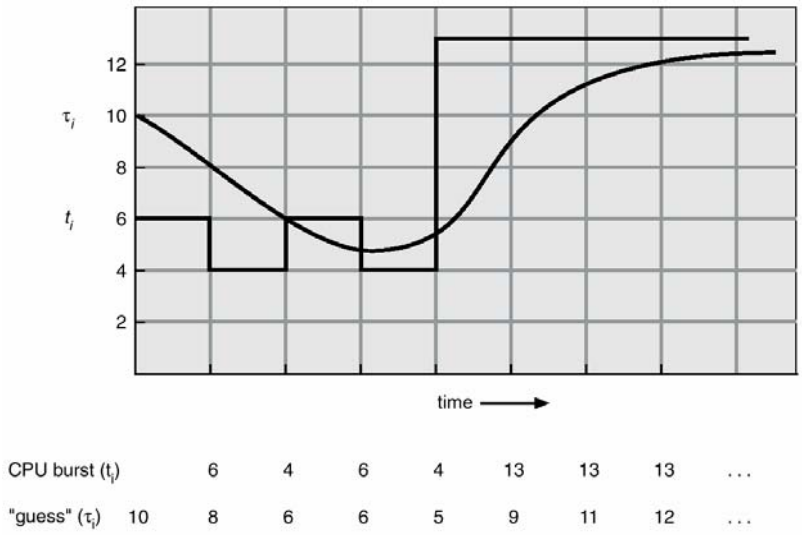
Để hiểu hành vi của giá trị trung bình dạng mũ, chúng ta có thể mở rộng công thức cho T_{n+1} bằng cách thay thế T_n để tìm

$T_{n+1}=\alpha\, t_n+(1-\alpha)\, \alpha\, t_{n-1}+\ldots+(1-\alpha)^j\, \alpha\, t_{n-j}+\ldots+(1-\alpha)^{n-1} T_0$

Vì cả hai α và (1- α) là nhỏ hơn hay bằng 1, mỗi số hạng tiếp theo có trọng số nhỏ hơn số hạng trước đó.

Giải thuật SJF có thể trưng dụng hoặc không trưng dụng CPU. Chọn lựa này phát sinh khi một quá trình mới đến tại hàng đợi sẵn sàng trong khi một quá trình trước đó đang thực thi. Một quá trình mới có thể có chu kỳ CPU tiếp theo ngắn hơn chu kỳ CPU được để lại của quá trình thực thi hiện tại. Giải thuật SJF trưng dụng sẽ trưng dungj CPU của quá trình đang thực thi hiện tại, trong khi giải thuật SJF không trưng dụng sẽ cho phép quá trình đang thực thi kết thúc chu kỳ CPU của nó. Định thời

SJF trưng dụng còn được gọi là **định thời thời gian còn lại ngắn nhất trước** (shortest-remaining-time-first).



Thí dụ, xét 4 quá trình sau với chiều dài của thời gian chu kỳ CPU được cho tính bằng mili giây

Quá trình	Thời gian đến	Thời gian xử lý
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Nếu các quá trình đi vào hàng đợi sẵn sàng tại những thời điểm và cần thời gian xử lý được hiển thị trong bảng trên thì thời biểu SJF trưng dụng được mô tả trong **lưu đồ Gannt** như sau:

P1	P2	P3		
0	1 5	10	17	26

Quá trình P_1 được bắt đầu tại thời điểm 0, vì nó là quá trình duy nhất trong hàng đợi. Quá trình P_2 đến tại thời điểm 1. Thời gian còn lại cho P_1 (7 mili giây) là lớn hơn thời gian được yêu cầu bởi quá trình P_2 (4 mili giây) vì thế quá trình P_1 bị trưng dụng CPU và quá trình P_2 được định thời biểu. Thời gian chờ đợi trung bình cho thí dụ này là: ((10-1) + (1-1) + (17-2) + (5-3))/4 = 6.5 mili giây. Định thời SJF không trưng dụng cho kết quả thời gian chờ đợi trung bình là 7.75 mili giây.

V.3 Định thời theo độ ưu tiên

Giải thuật SJF là trường hợp đặc biệt của **giải thuật định thời theo độ ưu tiên** (priority-scheduling algorithm). Độ ưu tiên được gán với mỗi quá trình và CPU được cấp phát tới quá trình với độ ưu tiên cao nhất. Quá trình có độ ưu tiên bằng nhau được định thời trong thứ tự FCFS.

Giải thuật SJF là giải thuật ưu tiên đơn giản ở đó độ ưu tiên **p** là nghịch đảo với chu kỳ CPU được đoán tiếp theo. Chu kỳ CPU lớn hơn có độ ưu tiên thấp hơn và ngược lại.

Bây giờ chúng ta thảo luận định thời có độ ưu tiên cao và thấp. Các độ ưu tiên thường nằm trong dãy số cố định, chẳng hạn 0 tới 7 hay 0 tới 4,095. Tuy nhiên, không có sự thoả thuận chung về 0 là độ ưu tiên thấp nhất hay cao nhất. Một vài hệ thống dùng số thấp để biểu diễn độ ưu tiên thấp; ngược lại các hệ thống khác dùng các số thấp cho độ ưu tiên cao. Sự khác nhau này có thể dẫn đến sự lẫn lộn. Trong giáo trình này chúng ta dùng các số thấp để biểu diễn độ ưu tiên cao.

Thí dụ, xét tập hợp quá trình sau đến tại thời điểm 0 theo thứ tự P_1, P_2,…, P_5 với chiều dài thời gian chu kỳ CPU được tính bằng mili giây:

Quá trình	Thời gian xử lý	Độ ưu tiên
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Sử dụng định thời theo độ ưu tiên, chúng ta sẽ định thời các quá trình này theo **lưu đồ Gannt** như sau:

P2	P5	P1	P3	P4
0	1 6	16	18	19

Thời gian chờ đợi trung bình là 8.2 mili giây.

Độ ưu tiên có thể được định nghĩa bên trong hay bên ngoài. Độ ưu tiên được định nghĩa bên trong thường dùng định lượng hoặc nhiều định lượng có thể đo để tính toán độ ưu tiên của một quá trình. Thí dụ, các giới hạn thời gian, các yêu cầu bộ nhớ, số lượng tập tin đang mở và tỉ lệ của chu kỳ nhập/xuất trung bình với tỉ lệ của chu kỳ CPU trung bình. Các độ ưu tiên bên ngoài được thiết lập bởi các tiêu chuẩn bên ngoài đối với hệ điều hành như sự quan trọng của quá trình, loại và lượng chi phí đang được trả cho việc dùng máy tính, văn phòng hỗ trợ công việc, ..

Định thời biểu theo độ ưu tiên có thể trưng dụng hoặc không trưng dụng CPU. Khi một quá trình đến hàng đợi sẵn sàng, độ ưu tiên của nó được so sánh với độ ưu tiên của quá trình hiện đang chạy. Giải thuật định thời theo độ ưu tiên trưng dụng sẽ chiếm CPU nếu độ ưu tiên của quá trình mới đến cao hơn độ ưu tiên của quá trình đang thực thi. Giải thuật định thời theo độ ưu tiên không trưng dụng sẽ đơn giản đặt quá trình mới tại đầu hàng đợi sẵn sàng.

Vấn đề chính với giải thuật định thời theo độ ưu tiên là **nghẽn không hạn định** (indefinite blocking) hay **đói CPU** (starvation). Một quá trình sẵn sàng chạy nhưng thiếu CPU có thể xem như bị nghẽn-chờ đợi CPU. Giải thuật định thời theo độ ưu tiên có thể để lại nhiều quá trình có độ ưu tiên thấp chờ CPU không hạn định. Trong một hệ thống máy tính tải cao, dòng đều đặn các quá trình có độ ưu tiên cao hơn có thể ngăn chặn việc nhận CPU của quá trình có độ ưu tiên thấp.. Thông thường, một trong hai trường hợp xảy ra. Cuối cùng, một quá trình sẽ được chạy (lúc 2 a.m chủ nhật là thời điểm cuối cùng hệ thống nạp các quá trình nhẹ), hay cuối cùng hệ thống máy tính sẽ đổ vỡ và mất tất cả các quá trình có độ ưu tiên thấp chưa được kết thúc.

Một giải pháp cho vấn đề nghẽn không hạn định này là **sự hoá già** (aging). Hóa già là kỹ thuật tăng dần độ ưu tiên của quá trình chờ trong hệ thống một thời gian dài. Thí dụ, nếu các độ ưu tiên nằm trong dãy từ 127 (thấp) đến 0 (cao), chúng ta giảm độ ưu tiên của quá trình đang chờ xuống 1 mỗi 15 phút. Cuối cùng, thậm chí một quá

trình với độ ưu tiên khởi đầu 127 sẽ đạt độ ưu tiên cao nhất trong hệ thống và sẽ được thực thi. Thật vậy, một quá trình sẽ mất không quá 32 giờ để đạt được độ ưu tiên từ 127 tới 0.

V.4 Định thời luân phiên

Giải thuật định thời luân phiên (round-robin scheduling algorithm-RR) được thiết kế đặc biệt cho hệ thống chia sẻ thời gian. Tương tự như định thời FCFS nhưng sự trưng dụng CPU được thêm vào để chuyển CPU giữa các quá trình. Đơn vị thời gian nhỏ được gọi là định mức thời gian (time quantum) hay phần thời gian (time slice) được định nghĩa. Định mức thời gian thường từ 10 đến 100 mili giây. Hàng đợi sẵn sàng được xem như một hàng đợi vòng. Bộ định thời CPU di chuyển vòng quanh hàng đợi sẵn sàng, cấp phát CPU tới mỗi quá trình có khoảng thời gian tối đa bằng một định mức thời gian.

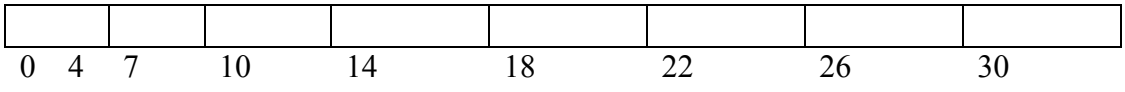
Để cài đặt định thời RR, chúng ta quản lý hàng đợi sẵn sàng như một hàng đợi FIFO của các quá trình. Các quá trình mới được thêm vào đuôi hàng đợi. Bộ định thời CPU chọn quá trình đầu tiên từ hàng đợi sẵn sàng, đặt bộ đếm thời gian để ngắt sau 1 định mức thời gian và gởi tới quá trình.

Sau đó, một trong hai trường hợp sẽ xảy ra. Quá trình có 1 chu kỳ CPU ít hơn 1 định mức thời gian. Trong trường hợp này, quá trình sẽ tự giải phóng. Sau đó, bộ định thời biểu sẽ xử lý quá trình tiếp theo trong hàng đợi sẵn sàng. Ngược lại, nếu chu kỳ CPU của quá trình đang chạy dài hơn 1 định mức thời gian thì độ đếm thời gian sẽ báo và gây ra một ngắt tới hệ điều hành. Chuyển đổi ngữ cảnh sẽ được thực thi và quá trình được đặt trở lại tại đuôi của hàng đợi sẵn sàng. Sau đó, bộ định thời biểu CPU sẽ chọn quá trình tiếp theo trong hàng đợi sẵn sàng.

Tuy nhiên, thời gian chờ đợi trung bình dưới chính sách RR thường là quá dài. Xét một tập hợp các quá trình đến tại thời điểm 0 với chiều dài thời gian CPU-burst được tính bằng mili giây:

Quá trình	Thời gian xử lý
P1	24
P2	3
P3	3

Nếu sử dụng định mức thời gian là 4 mili giây thì quá trình P_1 nhận 4 mili giây đầu tiên. Vì nó yêu cầu 20 mili giây còn lại nên nó bị trưng dụng CPU sau định mức thời gian đầu tiên và CPU được cấp tới quá trình tiếp theo trong hàng đợi, quá trình P_2. Vì P_2 không cần tới 4 mili giây nên nó kết thúc trước khi định mức thời gian của nó hết hạn. Sau đó, CPU được cho tới quá trình kế tiếp, quá trình P_3. Một khi mỗi quá trình nhận 1 định mức thời gian thì CPU trả về quá trình P_1 cho định mức thời gian tiếp theo. Thời biểu RR là:



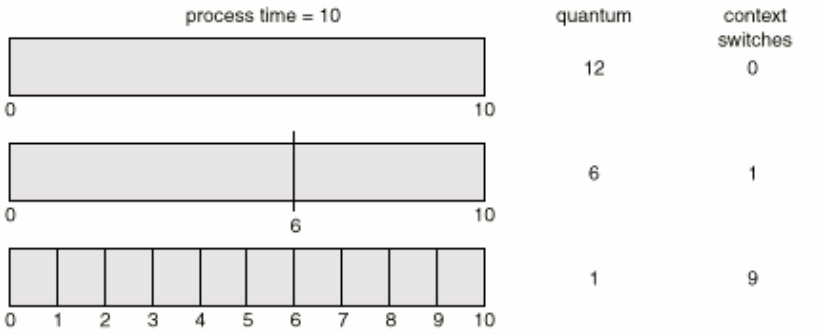
Thời gian chờ đợi trung bình là 17/3=5.66 mili giây.

Trong giải thuật RR, không quá trình nào được cấp phát CPU cho nhiều hơn 1 định mức thời gian trong một hàng. Nếu chu kỳ CPU của quá trình vượt quá 1 định

mức thời gian thì quá trình đó bị trưng dụng CPU và nó được đặt trở lại hàng đợi sẵn sàng. Giải thuật RR là giải thuật trưng dụng CPU.

Nếu có n quá trình trong hàng đợi sẵn sàng và định mức thời gian là q thì mỗi quá trình nhận 1/n thời gian CPU trong các phần, nhiều nhất q đơn vị thời gian. Mỗi quá trình sẽ chờ không dài hơn (n-1)x q đơn vị thời gian cho tới khi định mức thời gian tiếp theo của nó. Thí dụ, nếu có 5 quá trình với định mức thời gian 20 mili giây thì mỗi quá trình sẽ nhận 20 mili giây sau mỗi 100 mili giây.

Năng lực của giải thuật RR phụ thuộc nhiều vào kích thước của định mức thời gian. Nếu định mức thời gian rất lớn (lượng vô hạn) thì chính sách RR tương tự như chính sách FCFS. Nếu định mức thời gian là rất nhỏ (1 mili giây) thì tiếp cận RR được gọi là **chia sẻ bộ xử lý** (processor sharing) và xuất hiện (trong lý thuyết) tới người dùng như thể mỗi quá trình trong n quá trình có bộ xử lý riêng của chính nó chạy tại 1/n tốc độ của bộ xử lý thật.

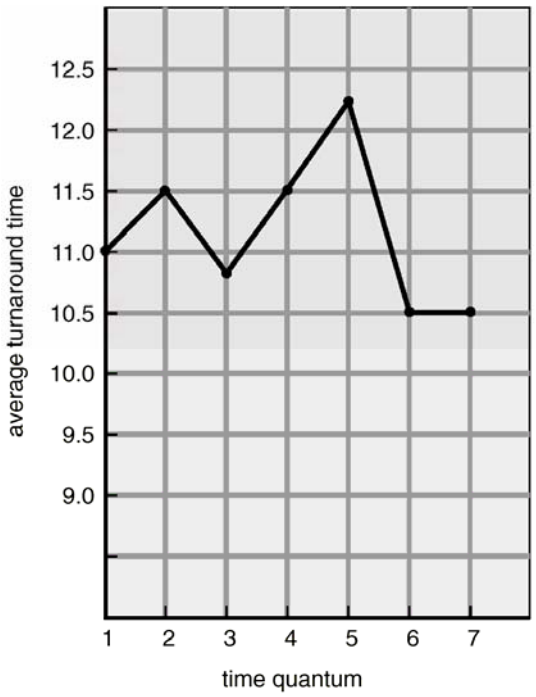


Hình 0-3 Hiển thị một định mức thời gian nhỏ hơn tăng chuyển đổi ngữ cảnh như thế nào

Tuy nhiên, trong phần mềm chúng ta cũng cần xem xét hiệu quả của việc chuyển đổi ngữ cảnh trên năng lực của việc định thời RR. Chúng ta giả sử rằng chỉ có 1 quá trình với 10 đơn vị thời gian. Nếu một định mức là 12 đơn vị thời gian thì quá trình kết thúc ít hơn 1 định mức thời gian, với không có chi phí nào khác. Tuy nhiên, nếu định mức là 6 đơn vị thời gian thì quá trình cần 2 định mức thời gian, dẫn đến 1 chuyển đổi ngữ cảnh. Nếu định mức thời gian là 1 đơn vị thời gian thì 9 chuyển đổi ngữ cảnh sẽ xảy ra, việc thực thi của quá trình bị chậm như được hiển thị trong hình IV.3 .

Do đó chúng ta mong muốn định mức thời gian lớn đối với thời gian chuyển ngữ cảnh. Nếu thời gian chuyển ngữ cảnh chiếm 10% định mức thời gian thì khoảng 10% thời gian CPU sẽ được dùng cho việc chuyển ngữ cảnh.

Thời gian hoàn thành cũng phụ thuộc kích thước của định mức thời gian. Chúng ta có thể thấy trong hình IV.4, thời gian hoàn thành trung bình của tập hợp các quá trình không cần cải tiến khi kích thước định mức thời gian tăng. Thông thường, thời gian hoàn thành trung bình có thể được cải tiến nếu hầu hết quá trình kết thúc chu kỳ CPU kế tiếp của chúng trong một định mức thời gian. Thí dụ, cho 3 quá trình có 10 đơn vị thời gian cho mỗi quá trình và định mức thời gian là 1 đơn vị thời gian, thì thời gian hoàn thành trung bình là 29. Tuy nhiên, nếu định mức thời gian là 10 thì thời gian hoàn thành trung bình giảm tới 20. Nếu thời gian chuyển ngữ cảnh được thêm vào thì thời gian hoàn thành trung bình gia tăng đối với định mức thời gian nhỏ hơn vì các chuyển đổi ngữ cảnh thêm nữa sẽ được yêu cầu.



process	time
P_1	6
P_2	3
P_3	1
P_4	7

Hình 0-4 Hiển thị cách thời gian hoàn thành biến đổi theo định mức thời gian

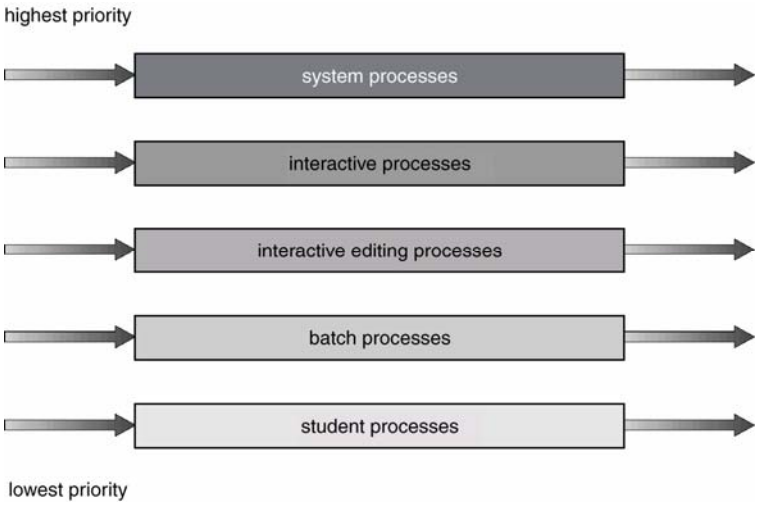
Ngoài ra, nếu định mức thời gian quá lớn thì người thiết kế việc định thời RR bao gồm chính sách FCFS. Qui tắc là định mức thời gian nên dài hơn 80% chu kỳ CPU.

V.5 Định thời biểu với hàng đợi nhiều cấp

Một loại giải thuật định thời khác được tạo ra cho những trường hợp mà trong đó các quá trình được phân lớp thành các nhóm khác nhau. Thí dụ: việc phân chia thông thường được thực hiện giữa các quá trình chạy ở chế độ giao tiếp (foreground hay interactive) và các quá trình chạy ở chế độ nền hay dạng bó (background hay batch). Hai loại quá trình này có yêu cầu đáp ứng thời gian khác nhau và vì thế có yêu cầu về định thời biểu khác nhau. Ngoài ra, các quá trình chạy ở chế độ giao tiếp có độ ưu tiên (hay được định nghĩa bên ngoài) cao hơn các quá trình chạy ở chế độ nền.

Một **giải thuật định thời hàng đợi nhiều cấp** (multilevel queue-scheduling algorithm) chia hàng đợi thành nhiều hàng đợi riêng rẻ (hình IV.5). Các quá trình được gán vĩnh viễn tới một hàng đợi, thường dựa trên thuộc tính của quá trình như kích thước bộ nhớ, độ ưu tiên quá trình hay loại quá trình. Mỗi hàng đợi có giải thuật định thời của chính nó. Thí dụ: các hàng đợi riêng rẻ có thể được dùng cho các quá trình ở chế độ nền và chế độ giao tiếp. Hàng đợi ở chế độ giao tiếp có thể được định thời bởi giải thuật RR trong khi hàng đợi ở chế độ nền được định thời bởi giải thuật FCFS.

Ngoài ra, phải có việc định thời giữa các hàng đợi, mà thường được cài đặt như định thời trưng dụng với độ ưu tiên cố định. Thí dụ, hàng đợi ở chế độ giao tiếp có độ ưu tiên tuyệt đối hơn hàng đợi ở chế độ nền.



Chúng ta xét một thí dụ của giải thuật hàng đợi nhiều mức với năm hàng đợi:

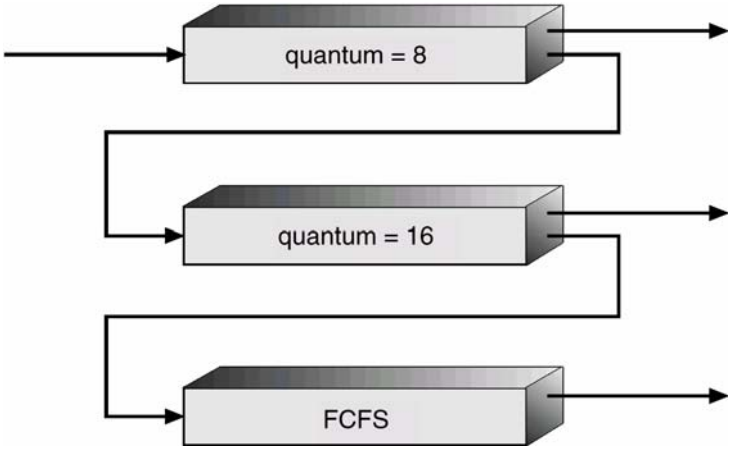
- Các quá trình hệ thống
- Các quá trình giao tiếp
- Các quá trình soạn thảo giao tiếp
- Các quá trình bó
- Các quá trình sinh viên

Mỗi hàng đợi có độ ưu tiên tuyệt đối hơn hàng đợi có độ ưu tiên thấp hơn. Thí dụ: không có quá trình nào trong hàng đợi bó có thể chạy trừ khi hàng đợi cho các quá trình hệ thống, các quá trình giao tiếp và các quá trình soạn thảo giao tiếp đều rỗng. Nếu một quá trình soạn thảo giao tiếp được đưa vào hàng đợi sẵn sàng trong khi một quá trình bó đang chạy thì quá trình bó bị trưng dụng CPU. Solaris 2 dùng dạng giải thuật này.

Một khả năng khác là phần (slice) thời gian giữa hai hàng đợi. Mỗi hàng đợi nhận một phần thời gian CPU xác định, sau đó nó có thể định thời giữa các quá trình khác nhau trong hàng đợi của nó. Thí dụ, trong hàng đợi giao tiếp-nền, hàng đợi giao tiếp được cho 80% thời gian của CPU cho giải thuật RR giữa các quá trình của nó, ngược lại hàng đợi nền nhận 20% thời gian CPU cho các quá trình của nó theo cách FCFS.

V.6 Định thời hàng đợi phản hồi đa cấp

Thông thường, trong giải thuật hàng đợi đa cấp, các quá trình được gán vĩnh viễn tới hàng đợi khi được đưa vào hệ thống. Các quá trình không di chuyển giữa các hàng đợi. Nếu có các hàng đợi riêng cho các quá trình giao tiếp và các quá trình nền thì các quá trình không di chuyển từ một hàng đợi này tới hàng đợi khác vì các quá trình không thay đổi tính tự nhiên giữa giao tiếp và nền. Cách tổ chức có ích vì chi phí định thời thấp nhưng thiếu linh động và có thể dẫn đến tình trạng "đói CPU".



Hình 0-6 Các hàng đợi phản hồi nhiều cấp

Tuy nhiên, **định thời hàng đợi phản hồi đa cấp** (multilevel feedback queue scheduling) cho phép một quá trình di chuyển giữa các hàng đợi. Ý tưởng là tách riêng các quá trình với các đặc điểm chu kỳ CPU khác nhau. Nếu một quá trình dùng quá nhiều thời gian CPU thì nó sẽ được di chuyển tới hàng đợi có độ ưu tiên thấp. Cơ chế này để lại các quá trình hướng nhập/xuất và các quá trình giao tiếp trong các hàng đợi có độ ưu tiên cao hơn. Tương tự, một quá trình chờ quá lâu trong hàng đợi có độ ưu tiên thấp hơn có thể được di chuyển tới hàng đợi có độ ưu tiên cao hơn. Đây là hình thức của sự hóa già nhằm ngăn chặn sự đói CPU.

Thí dụ, xét một bộ định thời hàng đợi phản hồi nhiều cấp với ba hàng đợi được đánh số từ 0 tới 2 (như hình IV.6). Bộ định thời trước tiên thực thi tất cả quá trình chứa trong hàng đợi 0. Chỉ khi hàng đợi 0 rỗng nó sẽ thực thi các quá trình trong hàng đợi 1. Tương tự, các quá trình trong hàng đợi 2 sẽ được thực thi chỉ nếu hàng đợi 0 và 1 rỗng. Một quá trình đến hàng đợi 1 sẽ ưu tiên hơn quá trình đến hàng đợi 2. Tương tự, một quá trình đến hàng đợi 0 sẽ ưu tiên hơn một quá trình vào hàng đợi 1.

Một quá trình đưa vào hàng đợi sẵn sàng được đặt trong hàng đợi 0. Một quá trình trong hàng đợi 0 được cho một định mức thời gian là 8 mili giây. Nếu nó không kết thúc trong thời gian này thì nó sẽ di chuyển vào đuôi của hàng đợi 1. Nếu hàng đợi 0 rỗng thì quá trình tại đầu của hàng đợi 1 được cho định mức thời gian là 16 mili giây. Nếu nó không hoàn thành thì nó bị chiếm CPU và được đặt vào hàng đợi 2. Các quá trình trong hàng đợi 2 được chạy trên cơ sở FCFS chỉ khi hàng đợi 0 và 1 rỗng.

Giải thuật định thời này cho độ ưu tiên cao nhất tới bất cứ quá trình nào với chu kỳ CPU 8 mili giây hay ít hơn. Một quá trình như thế sẽ nhanh chóng nhận CPU, kết thúc chu kỳ CPU của nó và bỏ đi chu kỳ I/O kế tiếp của nó. Các quá trình cần hơn 8 mili giây nhưng ít hơn 24 mili giây được phục vụ nhanh chóng mặc dù với độ ưu tiên thấp hơn các quá trình ngắn hơn. Các quá trình dài tự động rơi xuống hàng đợi 2 và được phục vụ trong thứ tự FCFS với bất cứ chu kỳ CPU còn lại từ hàng đợi 0 và 1.

Nói chung, một bộ định thời hàng đợi phản hồi nhiều cấp được định nghĩa bởi các tham số sau:

- Số lượng hàng đợi
- Giải thuật định thời cho mỗi hàng đợi
- Phương pháp được dùng để xác định khi nâng cấp một quá trình tới hàng đợi có độ ưu tiên cao hơn.
- Phương pháp được dùng để xác định khi nào chuyển một quá trình tới hàng đợi có độ ưu tiên thấp hơn.

- Phương pháp được dùng để xác định hàng đợi nào một quá trình sẽ đi vào và khi nào quá trình đó cần phục vụ.

Định nghĩa bộ định thời biểu dùng hàng đợi phản hồi nhiều cấp trở thành giải thuật định thời CPU phổ biến nhất. Bộ định thời này có thể được cấu hình để thích hợp với hệ thống xác định. Tuy nhiên, bộ định thời này cũng yêu cầu một vài phương tiện chọn lựa giá trị cho tất cả tham số để định nghĩa bộ định thời biểu tốt nhất. Mặc dù một hàng đợi phản hồi nhiều cấp là cơ chế phổ biến nhất nhưng nó cũng là cơ chế phức tạp nhất.

VI Định thời biểu đa bộ xử lý

Phần trên thảo luận chúng ta tập trung vào những vấn đề định thời biểu CPU trong một hệ thống với một bộ vi xử lý đơn. Nếu có nhiều CPU, vấn đề định thời tương ứng sẽ phức tạp hơn. Nhiều khả năng đã được thử nghiệm và như chúng ta đã thấy với định thời CPU đơn bộ xử lý, không có giải pháp tốt nhất. Trong phần sau đây, chúng ta sẽ thảo luận vắn tắt một số vấn đề tập trung về định thời biểu đa bộ xử lý. Chúng ta tập trung vào những hệ thống mà các bộ xử lý của nó được xác định (hay đồng nhất) trong thuật ngữ chức năng của chúng; bất cứ bộ xử lý nào sẵn có thì có thể được dùng để chạy bất quá trình nào trong hàng đợi. Chúng ta cũng cho rằng truy xuất bộ nhớ là đồng nhất (uniform memory access-UMA). Chỉ những chương trình được biên dịch đối với tập hợp chỉ thị của bộ xử lý được cho mới có thể được chạy trên chính bộ xử lý đó.

Ngay cả trong một bộ đa xử lý đồng nhất đôi khi có một số giới hạn cho việc định thời biểu. Xét một hệ thống với một thiết bị nhập/xuất được gán tới một đường bus riêng của một bộ xử lý. Các quá trình muốn dùng thiết bị đó phải được định thời biểu để chạy trên bộ xử lý đó, ngược lại thiết bị đó là không sẵn dùng.

Nếu nhiều bộ xử lý xác định sẵn dùng thì chia sẻ tải có thể xảy ra. Nó có thể cung cấp một hàng đợi riêng cho mỗi bộ xử lý. Tuy nhiên, trong trường hợp này, một bộ xử lý có thể rảnh với hàng đợi rỗng, trong khi bộ xử lý khác rất bận. Để ngăn chặn trường hợp này, chúng ta dùng một hàng đợi sẵn sàng chung. Tất cả quá trình đi vào một hàng đợi và được định thời biểu trên bất cứ bộ xử lý sẵn dùng nào.

Trong một cơ chế như thế, một trong hai tiếp cận định thời biểu có thể được dùng. Trong tiếp cận thứ nhất, mỗi bộ xử lý định thời chính nó. Mỗi bộ xử lý xem xét hàng đợi sẵn sàng chung và chọn một quá trình để thực thi. Nếu chúng ta có nhiều bộ xử lý cố gắng truy xuất và cập nhật một cấu trúc dữ liệu chung thì mỗi bộ xử lý phải được lập trình rất cẩn thận. Chúng ta phải đảm bảo rằng hai bộ xử lý không chọn cùng quá trình và quá trình đó không bị mất từ hàng đợi. Tiếp cận thứ hai tránh vấn đề này bằng cách đề cử một bộ xử lý như bộ định thời cho các quá trình khác, do đó tạo ra cấu trúc chủ-tớ (master-slave).

Một vài hệ thống thực hiện cấu trúc này từng bước bằng cách tất cả quyết định định thời, xử lý nhập/xuất và các hoạt động hệ thống khác được quản lý bởi một bộ xử lý đơn-một server chủ. Các bộ xử lý khác chỉ thực thi mã người dùng. Đa xử lý không đối xứng (asymmetric multiprocessing) đơn giản hơn đa xử lý đối xứng (symmetric multiprocessing) vì chỉ một quá trình truy xuất các cấu trúc dữ liệu hệ thống, làm giảm đi yêu cầu chia sẻ dữ liệu. Tuy nhiên, nó cũng không hiệu quả. Các quá trình giới hạn nhập/xuất có thể gây thắt cổ chai (bottleneck) trên một CPU đang thực hiện tất cả các hoạt động. Điển hình, đa xử lý không đối xứng được cài đặt trước trong một hệ điều hành và sau đó được nâng cấp tới đa xử lý đối xứng khi hệ thống tiến triển.

VII Định thời thời gian thực

Trong chương đầu chúng ta đã tìm hiểu tổng quan về hệ điều hành thời thực và thảo luận tầm quan trọng của nó. Ở đây, chúng ta tiếp tục thảo luận bằng cách mô tả các điều kiện thuận lợi định thời cần để hỗ trợ tính toán thời thực trong hệ thống máy tính đa mục đích.

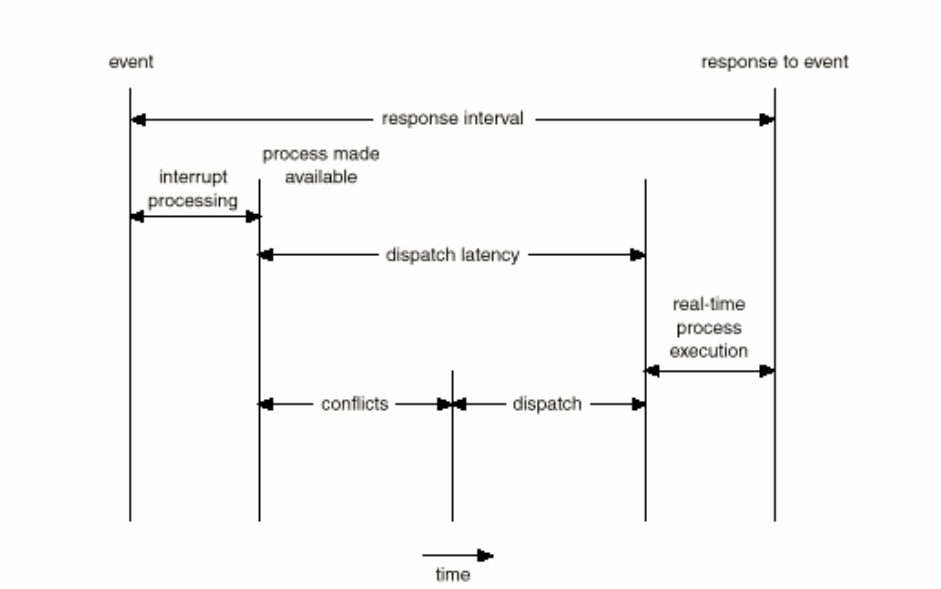
Tính toán thời thực được chia thành hai loại: hệ thống thời thực cứng (hardware real-time systems) được yêu cầu để hoàn thành một tác vụ tới hạn trong lượng thời gian được đảm bảo. Thông thường, một quá trình được đưa ra xem xét cùng với khai báo lượng thời gian nó cần để hoàn thành hay thực hiện nhập/xuất. Sau đó, bộ định thời biểu nhận được quá trình, đảm bảo rằng quá trình sẽ hoàn thành đúng giờ hay từ chối yêu cầu khi không thể. Điều này được gọi là đặt trước tài nguyên (resource reservation). Để đảm bảo như thế đòi hỏi bộ định thời biết chính xác bao lâu mỗi loại chức năng hệ điều hành mất để thực hiện và do đó mỗi thao tác phải được đảm bảo để mất lượng thời gian tối đa. Một đảm bảo như thế là không thể trong hệ thống với lưu trữ phụ và bộ nhớ ảo vì các hệ thống con này gây ra sự biến đổi không thể tránh hay không thể thấy trước trong lượng thời gian thực thi một quá trình xác định. Do đó, hệ thống thời thực cứng được hình thành từ nhiều phần mềm có mục đích đặc biệt chạy trên phần cứng tận hiến cho các quá trình tới hạn, và thiếu chức năng đầy đủ của các máy tính và các hệ điều hành hiện đại.

Tính toán thời gian thực mềm (soft real-time computing) ít nghiêm khắc hơn. Nó yêu cầu các quá trình tới hạn nhận độ ưu tiên cao hơn các quá trình khác. Mặc dù thêm chức năng thời thực mềm tới hệ chia sẻ thời gian có thể gây ra việc cấp phát tài nguyên không công bằng và có thể dẫn tới việc trì hoãn lâu hơn hay thậm chí đói tài nguyên đối với một số quá trình, nhưng nó ít có thể đạt được. Kết quả là hệ thống mục đích chung cũng có thể hỗ trợ đa phương tiện, đồ họa giao tiếp tốc độ cao, và nhiều tác vụ khác nhưng không hỗ trợ tính toán thời thực mềm.

Cài đặt chức năng thời thực mềm đòi hỏi thiết kế cẩn thận bộ định thời biểu và các khía cạnh liên quan của hệ điều hành. Trước tiên, hệ thống phải có định thời trưng dụng và các quá trình thời thực phải có độ ưu tiên cao nhất. Độ ưu tiên của các quá trình thời thực phải không giảm theo thời gian mặc dù độ ưu tiên của các quá trình không thời thực có thể giảm. Thứ hai, độ trễ của việc điều phối phải nhỏ. Một quá trình thời thực nhỏ hơn, nhanh hơn có thể bắt đầu thực thi một khi nó có thể chạy.

Quản trị các thuộc tính đã được xem xét ở trên là tương đối đơn giản. Thí dụ, chúng ta có thể không cho phép một quá trình hóa già trên các quá trình thời thực, do đó đảm bảo rằng độ ưu tiên của các quá trình không thay đổi. Tuy nhiên, đảm bảo thuộc tính sau đây phức tạp hơn. Vấn đề là nhiều hệ điều hành gồm hầu hết ấn bản của UNIX bị bắt buộc chờ lời gọi hệ thống hoàn thành hay nghẽn nhập/xuất xảy ra trước khi thực hiện chuyển ngữ cảnh. Độ trễ điều phối trong những hệ thống như thế có thể dài vì một số lời gọi hệ thống phức tạp và một vài thiết bị nhập/xuất chậm. Để giữ độ trễ điều phối chậm, chúng ta cần cho phép các lời gọi hệ thống được trưng dụng. Có nhiều cách để đạt mục đích này. Cách thứ nhất là chèn các điểm trưng dụng (preemption points) trong những lời gọi hệ thống có khoảng thời gian dài, kiểm tra để thấy quá trình ưu tiên cao cần được thực thi hay không. Nếu đúng, thì chuyển ngữ cảnh xảy ra và khi quá trình có độ ưu tiên kết thúc, quá trình bị ngắt tiếp tục với lời gọi hệ thống. Các điểm trưng dụng chỉ có thể được đặt tại vị trí "an toàn" trong nhân-nơi mà những cấu trúc dữ liệu hiện tại không được cập nhật. Ngay cả với độ trễ điều phối trưng dụng có thể lớn vì chỉ một vài điểm trưng dụng có thể được thêm vào nhân trong thực tế.

Một phương pháp khác để giải quyết sự trưng dụng là làm toàn bộ nhân có thể trưng dụng. Để đảm bảo các hoạt động thực hiện đúng, tất cả cấu trúc dữ liệu nhân phải được bảo vệ thông qua việc sử dụng các cơ chế đồng bộ hóa. Với phương pháp này, nhân luôn có thể trưng dụng vì bất cứ dữ liệu nhân được cập nhật được bảo vệ từ việc sửa đổi bởi quá trình có độ ưu tiên cao. Đây là một phương pháp hiệu quả nhất trong việc sử dụng rộng rãi; nó được dùng trong Solaris 2.



Hình 0-7 Độ trễ gửi

Nhưng điều gì xảy ra nếu quá trình có độ ưu tiên cao cần đọc hay sửa đổi dữ liệu nhân hiện đang được truy xuất bởi quá trình khác có độ ưu tiên thấp hơn? Quá trình có độ ưu tiên cao đang chờ quá trình có độ ưu tiên thấp kết thúc. Trường hợp này được gọi là đảo ngược độ ưu tiên (prioprity inversion). Thật vậy, một chuỗi các quá trình đang truy xuất tài nguyên mà quá trình có độ ưu tiên cao cần. Vấn đề này có thể giải quyết bằng giao thức kế thừa độ ưu tiên (priority-inheritance protocol) trong đó tất cả quá trình này (các quá trình này truy xuất tài nguyên mà quá trình có độ ưu tiên cao cần) kế thừa độ ưu tiên cao cho đến khi chúng được thực hiện với tài nguyên trong câu hỏi. Khi chúng kết thúc, độ ưu tiên của chúng chuyển trở lại giá trị ban đầu của nó.

Trong hình IV.7, chúng ta hiển thị sự thay đổi của độ trễ điều phối. Giai đoạn xung đột (conflict phase) của độ trễ điều phối có hai thành phần:

- Sự trưng dụng của bất cứ quá trình nào đang chạy trong nhân
- Giải phóng tài nguyên các quá trình có độ ưu tiên thấp được yêu cầu bởi quá trình có độ ưu tiên cao

Thí dụ, trong Solaris 2 độ trễ điều phối với sự trưng dụng bị vô hiệu hóa khi vượt qua 100 mili giây. Tuy nhiên, độ trễ điều phối với sự trưng dụng được cho phép thường được giảm xuống tới 2 mili giây.

VIII Đánh giá giải thuật

Chúng ta chọn một giải thuật định thời CPU cho một hệ thống xác định như thế nào? Có nhiều giải thuật định thời, mỗi giải thuật với các tham số của riêng nó. Do đó, chọn một giải thuật có thể là khó.

Vấn đề đầu tiên là định nghĩa các tiêu chuẩn được dùng trong việc chọn một giải thuật. Các tiêu chuẩn thường được định nghĩa trong thuật ngữ khả năng sử dụng CPU, thời gian đáp ứng hay thông lượng. Để chọn một giải thuật, trước hết chúng ta phải định nghĩa trọng số quan trọng của các thước đo này. Tiêu chuẩn của chúng ta có thể gồm các thước đo như:

- Khả năng sử dụng CPU tối đa dưới sự ràng buộc thời gian đáp ứng tối đa là 1 giây.
- Thông lượng tối đa như thời gian hoàn thành (trung bình) tỉ lệ tuyến tính với tổng số thời gian thực thi.

Một khi các tiêu chuẩn chọn lựa được định nghĩa, chúng ta muốn đánh giá các giải thuật khác nhau dưới sự xem xét. Chúng ta mô tả các phương pháp đánh giá khác nhau trong những phần dưới đây

VIII.1 Mô hình quyết định

Một loại quan trọng của phương pháp đánh giá được gọi là đánh giá phân tích (analytic evaluation). Đánh giá phân tích dùng giải thuật được cho và tải công việc hệ thống để tạo ra công thức hay số đánh giá năng lực của giải thuật cho tải công việc đó.

Một dạng đánh giá phân tích là **mô hình xác định** (deterministic modeling). Phương pháp này lấy tải công việc đặc biệt được xác định trước và định nghĩa năng lực của mỗi giải thuật cho tải công việc đó.

Thí dụ, giả sử rằng chúng ta có tải công việc được hiển thị trong bảng dưới. Tất cả 5 quá trình đến tại thời điểm 0 trong thứ tự được cho, với chiều dài của thời gian chu kỳ CPU được tính bằng mili giây.

Quá trình	Thời gian xử lý
P1	10
P2	29
P3	3
P4	7
P5	12

Xét giải thuật định thời FCFS, SJF và RR (định mức thời gian=10 mili giây) cho tập hợp quá trình này. Giải thuật nào sẽ cho thời gian chờ đợi trung bình tối thiểu?

Đối với giải thuật FCFS, chúng ta sẽ thực thi các quá trình này như sau:

P1	P2	P3	P4	P5
0 10	39	42	49	61

Thời gian chờ đợi là 0 mili giây cho quá trình P_1, 32 mili giây cho quá trình P_2, 39 giây cho quá trình P_3, 42 giây cho quá trình P_4 và 49 mili giây cho quá trình P_5. Do đó, thời gian chờ đợi trung bình là (0 + 10 + 39 + 42 + 49)/5= 28 mili giây.

Với định thời không trưng dụng SJF, chúng ta thực thi các quá trình như sau:

P3	P4	P1	P5	P2
0	3 10	20	32	61

Thời gian chờ đợi là 10 mili giây cho quá trình P_1, 32 mili giây cho quá trình P_2, 0 mili giây cho quá trình P_3, 3 mili giây cho quá trình P_4, và 20 giây cho quá trình P_5. Do đó, thời gian chờ đợi trung bình là (10 + 32 + 0 + 3 + 20)/5= 13 mili giây.

Với giải thuật RR, chúng ta thực thi các quá trình như sau:

P1	P2	P3	P4	P5	P2	P5	P2
0 10	20	23	30	40	50	52	61

Thời gian chờ đợi là 0 mili giây cho quá trình P_1, 32 mili giây cho quá trình P_2, 20 mili giây cho quá trình P_3, 23 mili giây cho quá trình P_4, và 40 mili giây cho quá trình P_5. Do đó, thời gian chờ đợi trung bình là (0 + 32 + 20 + 23 + 40)/5 = 23 mili giây.

Trong trường hợp này, chúng ta thấy rằng, chính sách SJF cho kết quả ít hơn ½ thời gian chờ đợi trung bình đạt được với giải thuật FCFS; giải thuật RR cho chúng ta giá trị trung gian.

Mô hình xác định là đơn giản và nhanh. Nó cho các con số chính xác, cho phép các giải thuật được so sánh với nhau. Tuy nhiên, nó đòi hỏi các số đầu vào chính xác và các trả lời của nó chỉ áp dụng cho những trường hợp đó. Việc dùng chủ yếu của mô hình xác định là mô tả giải thuật định thời và cung cấp các thí dụ. Trong các trường hợp, chúng ta đang chạy cùng các chương trình lặp đi lặp lại và có thể đo các yêu cầu xử lý của chương trình một cách chính xác, chúng ta có thể dùng mô hình xác định để chọn giải thuật định thời. Qua tập hợp các thí dụ, mô hình xác định có thể hiển thị khuynh hướng được phân tích và chứng minh riêng. Thí dụ, có thể chứng minh rằng đối với môi trường được mô tả (tất cả quá trình và thời gian của chúng sẵn dùng tại thời điểm 0), chính sách SJF sẽ luôn cho kết quả thời gian chờ đợi là nhỏ nhất.

Tuy nhiên, nhìn chung mô hình xác định quá cụ thể và yêu cầu tri thức quá chính xác để sử dụng nó một cách có ích.

VIII.2 Mô hình hàng đợi

Các quá trình được chạy trên nhiều hệ thống khác nhau từ ngày này sang ngày khác vì thế không có tập hợp quá trình tĩnh (và thời gian) để dùng cho mô hình xác định. Tuy nhiên, những gì có thể được xác định là sự phân bổ chu kỳ CPU và I/O. Sự phân bổ này có thể được đo và sau đó được tính xấp xỉ hay ước lượng đơn giản. Kết quả là một công thức toán mô tả xác suất của một chu kỳ CPU cụ thể. Thông thường, sự phân bổ này là hàm mũ và được mô tả bởi giá trị trung bình của nó. Tương tự, sự phân bổ thời gian khi các quá trình đến trong hệ thống-phân bổ thời gian đến-phải được cho.

Hệ thống máy tính được mô tả như một mạng các server. Mỗi server có một hàng đợi cho các quá trình. CPU là một server với hàng đợi sẵn sàng của nó, như là một hệ thống nhập/xuất với các hàng đợi thiết bị. Biết tốc độ đến và tốc độ phục vụ, chúng ta có thể tính khả năng sử dụng, chiều dài hàng đợi trung bình, thời gian chờ

trung bình,..Lĩnh vực nghiên cứu này được gọi là phân tích mạng hàng đợi (queueing-network analysis).

Thí dụ, gọi n là chiều dài hàng đợi trung bình (ngoại trừ các quá trình đang được phục vụ), gọi *W* là thời gian chờ đợi trung bình trong hàng đợi và λ là tốc độ đến trung bình cho các quá trình mới trong hàng đợi (chẳng hạn 3 quá trình trên giây). Sau đó, chúng ta mong đợi trong suốt thời gian *W* một quá trình chờ, λ x *W* các quá trình mới sẽ đến trong hàng đợi. Nếu hệ thống ở trong trạng thái đều đặn thì số lượng quá trình rời hàng đợi phải bằng số lượng quá trình đến. Do đó,

$$n = \lambda \times W$$

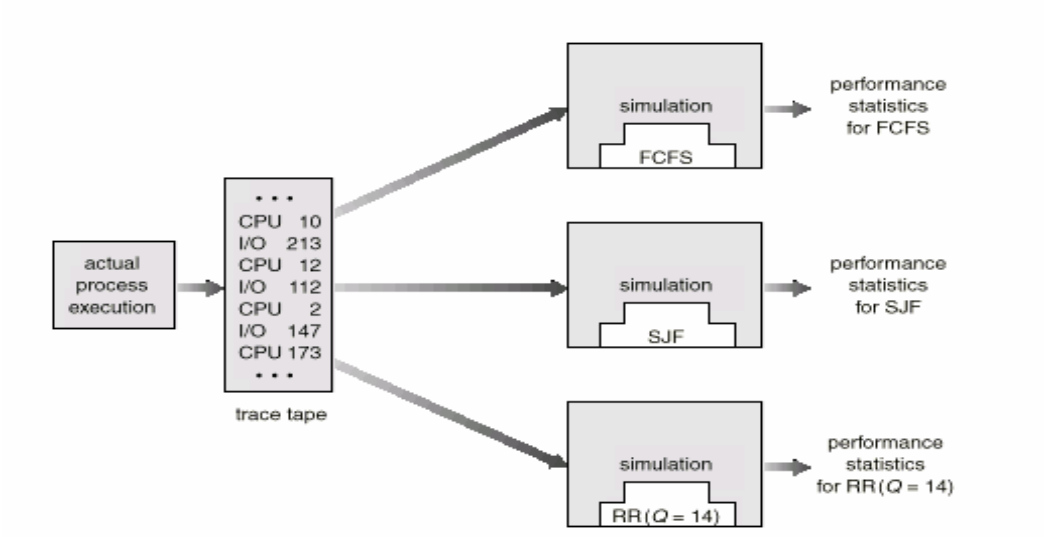
Công thức này được gọi là công thức Little. Công thức Little là đặc biệt có ích vì nó phù hợp cho bất cứ giải thuật định thời và sự phân bổ các quá trình đến.

Chúng ta sử dụng công thức Little để tính một trong ba biến, nếu chúng ta biết hai biến khác. Thí dụ, nếu chúng ta biết có 7 quá trình đến mỗi giây (trung bình) và thường có 14 quá trình trong hàng đợi thì chúng ta có thể tính thời gian chờ đợi trung bình trên mỗi quá trình là 2 giây.

Phân tích hàng đợi có thể có ích trong việc so sánh các giải thuật định thời nhưng nó cũng có một số giới hạn. Hiện nay, các loại giải thuật và sự phân bổ được quản lý là tương đối giới hạn. Tính toán của các giải thuật phức tạp và sự phân bổ là rất khó để thực hiện. Do đó, phân bổ đến và phục vụ thường được định nghĩa không thực tế, nhưng dễ hướng dẫn về mặt tính toán. Thông thường cần thực hiện một số giả định độc lập có thể không chính xác. Do đó, để chúng sẽ có thể tính câu trả lời, các mô hình hàng đợi thường chỉ xấp xỉ hệ thống thật. Vì thế, độ chính xác của các kết quả tính toán có thể là sự nghi vấn.

VIII.3 Mô phỏng

Để đạt được sự đánh giá các giải thuật định thời chính xác hơn, chúng ta có thể dùng mô phỏng (simulations). Mô phỏng liên quan đến lập trình một mô hình hệ thống máy tính. Cấu trúc dữ liệu phần mềm biểu diễn các thành phần quan trọng của hệ thống. Bộ mô phỏng có một biến biểu diễn đồng hồ; khi giá trị của biến này tăng, bộ mô phỏng sửa đổi trạng thái hệ thống để phản ánh các hoạt động của các thiết bị, các quá trình và các bộ định thời. Khi sự mô phỏng thực thi, các thống kê hiển thị năng lực của giải thuật được tập hợp và in ra.



Hình 0-8 Đánh giá các bộ định thời CPU bằng mô phỏng

Dữ liệu để định hướng sự mô phỏng có thể được sinh ra trong nhiều cách. Cách thông dụng nhất dùng bộ sinh số ngẫu nhiên, được lập trình để sinh ra các quá trình, thời gian chu kỳ CPU, đến, đi của quá trình,..dựa trên phân bổ xác suất. Sự phân bổ này có thể được định nghĩa dạng toán học (đồng nhất, hàm mũ, phân bổ Poisson) hay theo kinh nghiệm. Nếu sự phân bổ được định nghĩa theo kinh nghiệm thì các thước đo của hệ thống thật dưới sự nghiên cứu là lấy được. Các kết quả được dùng để định nghĩa sự phân bổ thật sự các sự kiện trong hệ thống thực và sau đó sự phân bổ này có thể được dùng để định hướng việc mô phỏng.

Tuy nhiên, một mô phỏng hướng phân bổ có thể không chính xác do mối quan hệ giữa các sự kiện tiếp theo trong hệ thống thực. Sự phân bổ thường xuyên hiển thị chỉ bao nhiêu sự kiện xảy ra; nó không hiển thị bất cứ thứ gì về thứ tự xảy ra của chúng. Để sửa chữa vấn đề này, chúng ta dùng băng từ ghi vết (trace tapes). Chúng ta tạo một băng từ ghi vết bằng cách giám sát hệ thống thực, ghi lại chuỗi các sự kiện thật (như hình IV.8). Sau đó, thứ tự này được dùng để định hướng việc mô phỏng. Băng từ ghi vết cung cấp cách tuyệt vời để so sánh chính xác hai giải thuật trên cùng một tập hợp dữ liệu vào thật. Phương pháp này có thể cung cấp các kết quả chính xác cho dữ liệu vào của nó.

Tuy nhiên, mô phỏng có thể rất đắt và thường đòi hỏi hàng giờ máy tính để thực hiện. Một mô phỏng chi tiết hơn cung cấp các kết quả chính xác hơn nhưng cũng yêu cầu nhiều thời gian máy tính hơn. Ngoài ra, các băng từ ghi vết có thể yêu cầu lượng lớn không gian lưu trữ. Cuối cùng, thiết kế, mã, gỡ rối của bộ mô phỏng là một tác vụ quan trọng.

VIII.4 Cài đặt

Ngay cả mô phỏng cũng cho độ chính xác có giới hạn. Chỉ có cách chính xác hoàn toàn để đánh giá giải thuật định thời là mã hóa (code) nó, đặt nó vào trong hệ điều hành và xem nó làm việc như thế nào. Tiếp cận này đặt một giải thuật thật sự vào hệ thống thật để đánh giá dưới điều kiện hoạt động thật sự.

Khó khăn chủ yếu là chi phí của tiếp cận. Chi phí bao gồm không chỉ mã hóa giải thuật và sửa đổi hệ điều hành để hỗ trợ nó cũng như các cấu trúc dữ liệu được yêu cầu mà còn phản ứng của người dùng đối với sự thay đổi liên tục hệ điều hành. Hầu hết người dùng không quan tâm việc xây dựng một hệ điều hành tốt hơn; họ chỉ đơn thuần muốn biết các quá trình của họ thực thi và dùng các kết quả của chúng. Một hệ điều hành thay đổi liên tục không giúp cho người dùng nhận thấy công việc của họ được thực hiện. Một dạng của phương pháp này được dùng phổ biến cho việc cài đặt máy tính mới. Thí dụ, một tiện ích Web mới có thể mô phỏng tải người dùng được phát sinh trước khi nó "sống" (goes live), để xác định bất cứ hiện tượng thắt cổ chai trong tiện ích và để ước lượng bao nhiêu người dùng hệ thống có thể hỗ trợ.
Một khó khăn khác với bất cứ việc đánh giá giải thuật nào là môi trường trong đó giải thuật được dùng sẽ thay đổi. Môi trường sẽ thay đổi không chỉ trong cách thông thường như những chương trình mới được viết và các loại vấn đề thay đổi, mà còn kết quả năng lực của bộ định thời. Nếu các quá trình được cho với độ ưu tiên ngắn thì người dùng có thể tách các quá trình lớn thành tập hợp các quá trình nhỏ hơn. Nếu quá trình giao tiếp được cho độ ưu tiên vượt qua các quá trình không giao tiếp thì người dùng có thể chuyển tới việc dùng giao tiếp.

Thí dụ, trong DEC TOPS-20, hệ thống được phân loại các quá trình giao tiếp và không giao tiếp một cách tự động bằng cách xem lượng nhập/xuất thiết bị đầu cuối. Nếu một quá trình không có nhập hay xuất tới thiết bị đầu cuối trong khoảng thời gian 1 phút thì quá trình được phân loại là không giao tiếp và được di chuyển tới

hàng đợi có độ ưu tiên thấp. Chính sách này dẫn đến trường hợp một người lập trình sửa đổi chương trình của mình để viết một ký tự bất kỳ tới thiết bị đầu cuối tại khoảng thời gian đều đặn ít hơn 1 phút. Hệ thống này cho những chương trình này có độ ưu tiên cao mặc dù dữ liệu xuất của thiết bị đầu cuối là hoàn toàn không có ý nghĩa. Các giải thuật có khả năng mềm dẻo nhất có thể được thay đổi bởi người quản lý hay người dùng. Trong suốt thời gian xây dựng hệ điều hành, thời gian khởi động, thời gian chạy, các biến được dùng bởi các bộ định thời có thể được thay đổi để phản ánh việc sử dụng của hệ thống trong tương lai. Yêu cầu cho việc định thời biểu mềm dẻo là một trường hợp khác mà ở đó sự tách riêng các cơ chế từ chính sách là có ích. Thí dụ, nếu các hóa đơn cần được xử lý và in lập tức nhưng thường được thực hiện như công việc bó có độ ưu tiên thấp, hàng đợi bó được cho tạm thời độ ưu tiên cao hơn. Tuy nhiên, rất ít hệ điều hành chấp nhận loại định thời này.

IX Tóm tắt

Định thời CPU là một tác vụ chọn một quá trình đang chờ từ hàng đợi sẵn sàng và cấp phát CPU tới nó. CPU được cấp phát tới quá trình được chọn bởi bộ cấp phát. Định thời đến trước, được phục vụ trước (FCFS) là giải thuật định thời đơn giản nhất, nhưng nó có thể gây các quá trình ngắn chờ các quá trình quá trình quá dài. Định thời ngắn nhất, phục vụ trước (SJF) có thể tối ưu, cung cấp thời gian chờ đợi trung bình ngắn nhất. Cài đặt định thời SJF là khó vì đoán trước chiều dài của chu kỳ CPU kế tiếp là khó. Giải thuật SJF là trường hợp đặc biệt của giải thuật định thời trưng dụng thông thường. Nó đơn giản cấp phát CPU tới quá trình có độ ưu tiên cao nhất. Cả hai định thời độ ưu tiên và SJF có thể gặp phải trở ngại của việc đói tài nguyên.

Định thời quay vòng (RR) là hợp lý hơn cho hệ thống chia sẻ thời gian. Định thời RR cấp phát CPU tới quá trình đầu tiên trong hàng đợi sẵn sàng cho q đơn vị thời gian, ở đây q là định mức thời gian. Sau q đơn vị thời gian, nếu quá trình này không trả lại CPU thì nó bị chiếm và quá trình này được đặt vào đuôi của hàng đợi sẵn sàng. Vấn đề quan trọng là chọn định mức thời gian. Nếu định mức quá lớn, thì định thời RR giảm hơn định thời FCFS ; nếu định mức quá nhỏ thì chi phí định thời trong dạng thời gian chuyển ngữ cảnh trở nên thừa.

Giải thuật FCFS là không ưu tiên; giải thuật RR là ưu tiên. Các giải thuật SJF và ưu tiên có thể ưu tiên hoặc không ưu tiên.

Các giải thuật hàng đợi nhiều cấp cho phép các giải thuật khác nhau được dùng cho các loại khác nhau của quá trình. Chung nhất là hàng đợi giao tiếp ở chế độ hiển thị dùng định thời RR và hàng đợi bó chạy ở chế độ nền dùng định thời FCFS. Hàng đợi phản hồi nhiều cấp cho phép các quá trình di chuyển từ hàng đợi này sang hàng đợi khác.

Vì có nhiều giải thuật định thời sẵn dùng, chúng ta cần các phương pháp để chọn giữa chúng. Các phương pháp phân tích dùng cách thức phân tích toán học để xác định năng lực của giải thuật. Các phương pháp mô phỏng xác định năng lực bằng cách phỏng theo giải thuật định thời trên những mẫu ‘đại diện’ của quá trình và tính năng lực kết quả.